Proceedings of the 34th
Conference on Decision & Control
New Orleans, LA - December 1995

Control Problems and the Polynomial Time Hierarchy

Jorge E. Tierno*
Electrical Engineering 116-81
California Institute of Technology
Pasadena, CA 91125, U.S.A.

John C. Doyle
Control and Dynamical Systems
California Institute of Technology
Pasadena, CA 91125, U.S.A.

Keywords: complexity, stabilizability, polynomial hierarchy

Abstract

We classify control problems by exhibiting their alternating quantifier structure. This classification allows us to relate these control problems to the computational complexity classes of the Polynomial Time Hierarchy. A specific synthesis problem for uncertain systems is shown to be hard in the class Π_2^p.

1 Introduction

A large number of results establishing the computational complexity of control problems have been established in the last decade. A diverse collection of analysis, synthesis and model validation problems have been proven to be NP-Hard. We will show in this paper how the different problems arising in control systems analysis can be naturally classified according to their complexity by exhibiting their "alternating quantifier" structure.

This classification falls in line with the polynomial time hierarchy of complexity classes as defined in [2]. By placing a specific problem in one of the complexity classes of the polynomial hierarchy, we know what the worst case complexity of the problem is. Depending on the structure imposed on both the known and the uncertain or unknown parts of the problem, it will or will not be hard on its class. For one quantifier problems, there is now a clear picture of which problems are easy and which are hard. In general terms, more structure in the known part of the system makes its solution easier. Conversely, the more structured the unknown part of the system is, the harder its solution becomes.

In this paper we will present a first step in extending this picture to the class of two quantifier control problems. We present in this paper also one such proof. We will consider the design of a scheduled structured controllers for systems subject to sensor failing or "on-off" uncertainty. We prove this problem to be hard in the class Π_2^p. We will compare this result with the design of Linear Parameter Varying (LPV) controllers [9]. These two problems have the same two quantifier structure. However the former is hard in Π_2^p, and the latter reduces to a convex optimization problem.

2 The Polynomial Time Hierarchy

We give in this section a brief and intuitive introduction to the Polynomial Time Hierarchy complexity classes. A rigorous treatment of the subject can be found in [2] or [6]. Consider the standard satisfiability problem of boolean clauses:

$$\exists(x_1, x_2, \cdots, x_n) \text{ such that } p(x_1, x_2, \cdots, x_n) \; ? \quad (1)$$

where p is any boolean clause in the x_i variables. This is a standard NP-complete problem. Correspondingly, the complementary question will be complete in co-NP:

$$\forall(x_1, x_2, \cdots, x_n), \; \neg p(x_1, x_2, \cdots, x_n) \; ? \quad (2)$$

Note that the universal quantifier in the co-NP problem does not allow us to "verify" a solution in polynomial time, since in principle we would have to check all possible combinations of the x_i. However since the problem and its dual are equivalent the verification can be done in the dual problem. In this case usage of one or the other quantifier doesn't change the nature of the problem, or its complexity.

But, if both quantifiers are present in the same problem, we will not be able to do the polynomial time verification of a solution, in either the direct or dual problem. The nature of problems with two quantifiers is intuitively different than the nature of NP or co-NP complete problems. Though it has not yet been possible to prove that they are harder (in the standard polynomial time reduction sense), experts in the field believe them to be, and have devoted considerable effort into characterizing them.

In this paper we will define the different classes in that characterization, the Polynomial Time Hierarchy, by giving a complete problem in each class. Although a more abstract definition can be given, we believe that in this way we capture the essential characteristic of the complexity problem and present it at a level accessible for the general control audience. Once again the definition is made in general terms.

Definition 1 *Let Σ_k be the class of problems reducible in polynomial time to the the following feasibility question:*

$$\exists x_1, \forall x_2, \exists \cdots Q x_k \quad \textit{such that} \quad p(x_1, x_2, \cdots, x_k) \quad (3)$$

where the x_i are sets of boolean variables, and the quantifier Q is existential for k odd and universal otherwise.

Note that with this definition we will have:

$$\Sigma_1 = NP \quad (4)$$

so for completeness we will add the convention:

$$\Sigma_0 = P \quad (5)$$

Analogously the corresponding dual classes can be defined:

Definition 2 *Let Π_k be the class of problems reducible in polynomial time to the the following feasibility question:*

$$\forall x_1, \exists x_2, \forall \cdots Q x_k \quad \textit{such that} \quad \neg p(x_1, x_2, \cdots, x_k) \quad (6)$$

*E-Mail: jorge@hot.caltech.edu

where the x_i are sets of boolean variables, and the quantifier Q is existential for k even and universal otherwise.

We will also make the convention:

$$\Pi_0 = P \tag{7}$$

It is not known if there are finitely many distinct classes in the hierarchy.However some partial results exist that give some idea of the possible situations. We quote the following

Theorem 1 *[2] Either for all $k \geq 0$, $\Sigma_k \neq \Sigma_{k+1}$, or the polynomial hierarchy consists of finitely many different classes.*

3 Alternating Quantifiers in Control Theory

In this section we will present a classification of different control theory problems in terms of a basic problem preceeded by alternating quantifiers. This classification serves a double purpose. First it allows us to identify which problems are essentially similar and suitable to be solved by unified algorithms. In second place it will also allow us to give certain "bounds" in the hardness of the computation of their solutions; by placing a problem in a given complexity class we know that it's solution will not be harder that the solution of a canonical problem hard in the class.

The canonical analysis problem.

We will classify all our problem starting from a given canonical analysis or performance problem. We thus assume that there is a certain property of a system p_S that can be computed and that determines whether the given system meets the required specification. For most relevant performance specification of linear systems, the corresponding p_S is usually relatively simple to compute. Stability, $\mathcal{H}_2$ norm or the $\mathcal{H}_\infty$ norm for finite dimensional linear system can be computed in polynomial time (to a given accuracy). We will measure the complexity of all other problems relative to this one.

Synthesis and robustness analysis.

The next level in the hierarchy of problems includes both a synthesis and an analysis problem. Assume first that we are allowed t o design C a part of the system S. Assume also that we are required to design C in a given class $\mathbf{C}$. The performance condition becomes now a function of both C and S; we will use the notation $P_S(C)$. The corresponding synthesis problem then becomes:

$$\exists C \in \mathbf{C} \text{ such that } p_S(C) \tag{8}$$

Now suppose that the system depends on a parameter Δ, and that all the information we have about Δ is that it belongs to a pre-specified set $\mathbf{\Delta}$. In order to guarantee that the system always meets the performance condition we will have to answer the following question:

$$\forall \Delta \in \mathbf{\Delta} \text{ we have } p_S(C) \ ? \tag{9}$$

The computational complexity of this problems depends of course on the nature of the sets $\mathbf{C}$ and $\mathbf{\Delta}$, and on how C and Δ modify the system S.

Robust and scheduled controller design.

The next class of problems will contain two alternating quantifiers and, as was the case in the polynomial time hierarchy, they can be derived from the two problems previously stated. First consider the Robust controller design. We are required to design a controller such that the performance is met for all the systems in a class. Using the notation from the last section this can be written as the following question:

$$\exists C \in \mathbf{C} \text{ such that } \forall \Delta \in \mathbf{\Delta} \ p_S(C, \Delta) \ ? \tag{10}$$

The gain scheduled controller design problem consists of designing a controller for each instance of a set of parameters δ,that will achieve the required performance.Note that nothing is specified about the nature of the parameters δ. Different problems will give rise to different sets on which to allow δ to vary. They could be constants, functions of time or $\mathcal{L}_2$ operators. The essence of the gain scheduled controller is to allow the controller to depend on the parameter. A given performance level can thus be achieved by a gain scheduled controller if and only if:

$$\forall \delta \in \delta \ \exists C \in \mathbf{C} \text{ such that } p_S(C, \delta) \ ? \tag{11}$$

4 Hard Problems

In the previous section we presented a classification of several interesting control problems, according to the number of alternating quantifiers preceeding a feasibility question. However, not all problems in those classes are equally hard. Several robust analysis problems ([7],[10]), and gain scheduled synthesis problems ([9]) have been proven to be solvable in polynomial time. However, other problems have been proven to be hard in their respective classes. For example the μ analysis problem [4], output feedback synthesis [3], and certain model validation problems [11], have been proven to be hard in the first class of our classification corresponding to the first level (NP, co-NP) in the polynomial hierarchy. Although the general robust and gain scheduled controller synthesis problems are recognized to be much harder that their corresponding NP-complete problems, no systematic proofs of this are available. (Although some versions of the problem have been proven to be in P.) In this paper we present a first attempt at pinning down the hardness of those problems by proving that a specific case is Π_2 hard.

A Π_2 hard quadratic problem.

As it was done with the proof of NP-completeness of certain μ-analysis problems, we will begin by exhibiting a quadratic programming problem that is hard in the Π_2 class. In what follows we will consider the binary Turing machine computation model. The following problem is known to be NP-hard [8]:

$$\exists x \in Q, b \leq x \leq B | x^t A x + bx + c | \leq k \ ? \tag{12}$$

where inequalities are to be read element by element. The proof is done by reducing the Knapsack problem into the given quadratic programming problem.

Consider the following decision problem

$$\forall y \in \{0,1\}^m \ \exists x \in \{0,1\}^n \text{ such that } \sum_1^m d_j y_j + \sum_1^n d_j n_j - d_0 = 0 \ ? \tag{13}$$

a generalization of the problem referred to in the computational complexity literature as the partial sums problem. ([8]). The answer to the question is yes if and only if the answer to

$$\forall y \in \{0,1\}^m \ \exists x \in \{0,1\}^n \ \text{such that}$$
$$(\sum_1^m d_j y_j + \sum_1^n d_j n_j - d_0)^2 +$$
$$\sum_1^m y_j(1-y_j) + \sum_1^n x_j(1-x_j) \leq 0 \ ? \qquad (14)$$

is also yes. And since the inequality can only be satisfied when the left hand side is zero, we can relax the restriction on x and allow it to vary in the set $[0,1]^n$. The preceeding decision problems are then equivalent to:

$$\forall y \in \{0,1\}^m \ \exists x \in [0,1]^n \ \text{such that}$$
$$(\sum_1^m d_j y_j + \sum_1^n d_j n_j - d_0)^2 +$$
$$\sum_1^m y_j(1-y_j) + \sum_1^n x_j(1-x_j) \leq 0 \ ? \qquad (15)$$

Finally by expanding the squares and collecting terms according to their degree, we can see that the preceeding problems are equivalent to:

$$\forall y \in \{0,1\}^m \ \exists x \in [0,1]^n \ \text{such that}$$
$$|[yx]A[yx]^t + b[yx]^t + c| \leq k? \qquad (16)$$

for an appropriate choice of the matrix A the vector b and the constants c and k. The size of this elements is a polynomial function of n and m. This means that an instance of the original question can be answered by answering an instance of the final problem, with size a polynomial function of the original, and thus that the final decision problem is at least as hard as the original one. In what follows we will show that the original problem is Π_2 hard, and that the final problem can be interpreted as a structured scheduled controller problem.

Theorem 2 *The decision problem in (13) is Π_2 hard.*

Proof: The proof follows very closely the NP-hardness proof of the Subset Sum problem. We will show that we can reduce the Π_2 complete satisfiability problem:

$$\forall y \ \exists x \text{ such that } f([yx]) \ ? \qquad (17)$$

where f is a general boolean expression in the variables x and y. The reduction of the satisfiability problem to the Subset Sum problem follows the following links [5]: Satisfiability (SAT) $\rightarrow$ 3-Satisfiability (3SAT) $\rightarrow$ 3 Dimensional Matching (3DM)$\rightarrow$ Partition (PART) $\rightarrow$ Subset Sum (SS). We need to show that the same path can be used with the two alternating quantifiers versions of these problems. In this case to reduce problem P_1 to P_2, we have to prove that for every instance p_1 of P_1, we can build in polynomial time an instance p_2 of problem P_2 such that

$$\forall\ y_1 \in \mathcal{Y}_1 \ \exists x_1 \text{ such that } p_1(y_1, x_1)$$

is equivalent to

$$\forall\ y_2 \in \mathcal{Y}_2 \ \exists x_2 \text{ such that } p_1(y_2, x_2).$$

We will refer to the proofs and notations used in [5] to show how this can be done. Although this will make the exposition not self contained, the space limitations do not allow us to include the complete proofs of all the transformations. In the first reduction from SAT to 3-SAT, it is shown that

$$\exists x \ \ f(x) \Leftrightarrow \exists x, \exists z \ \ f_3([xz]) \qquad (18)$$

where f_3 is a boolean formula made of sum of products of three elements on the same variables x plus an additional set of variables z. Partition the set of variables x into two parts $[yx]$. Then the same reduction can be used to prove that:

$$\forall y \ \exists x \ \ f(x) \Leftrightarrow \forall y \ \exists x, \exists z \ f_3([yxz]) \qquad (19)$$

the second problem is an instance of the two quantifier 3SAT problem and thus the first reduction is complete. The second reduction takes 3SAT into 3DM. In this reduction each choice of values for the variables x in the 3SAT problem corresponds to a subset M' of a certain set M. (The reader is once again referred to [5] for a description of the notation used.) The choice of variables satisfies the 3SAT formula if and only if the subset M' can be completed to a matching set. In particular, all assignments of a partial set of variables y can be completed to satisfy the 3SAT formula, if and only if all the corresponding subsets can be completed to a matching set. Partition the variables in the 3-SAT problem into two sets y and x. Denote by M_y the subset of M corresponding to the assignment of variables y, by M_x the subset of M corresponding to the assignment of variables x and by $\mathcal{Y} = \{M_y : y \in \{0,1\}^n\}$. Then

$$\forall y \ \exists x \text{ such that } f_3([yx])$$
$$\Leftrightarrow$$
$$\forall M_y \in \mathcal{Y} \ \exists M_x \subset M \ \exists M_z \subset M \qquad (20)$$
$$\text{such that}$$
$$M' \cup M_x \cup M_z \text{ is a matching set of } M.$$

The reduction of 3DM to PART associates each element of M with a variable in the partition set. Correspondingly, each subset of M is mapped to a particular assignment of those variables, and it is shown that a subset M' of M can be completed to a matching set if and only if the corresponding assignment of variables can be completed to a partition. Finally the conversion of PART to SS is immediate. A partition exists if and only if a subset with sum equal to half the sum of all the weights exists. We have thus shown that the question in (17) can be answered by answering a question of the form (13) where the number of variables in the latter is a polynomial function of the number of variables in the former. Since (13) can in turn be reduced in polynomial time to (16) the theorem follows. ■

A Π_2 hard synthesis problem.

The problem posed in (16) has the structure of the scheduled controller problems described in Section 3. In the next theorem we will present a scheduled controller design problem that is Π_2 hard as a consequence of Theorem 2. Although the problem presented is restricted and somewhat artificial, we believe most of the restrictions can be lifted. We present it here as a first step in the analysis of the complexity of two alternating quantifiers control problem.

Theorem 3 *Consider a linear system of the form given in Figure 1, where δ_i can be 0 or 1. Answering the question: For each allowable instance of the parameters δ, does*

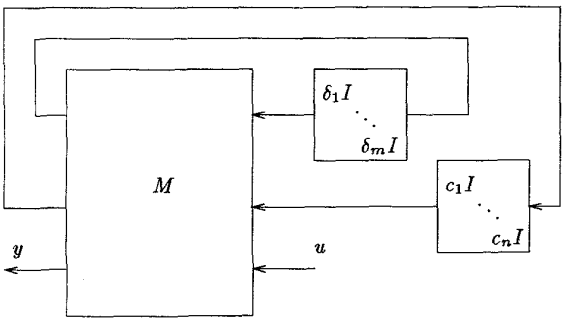

Figure 1: Sensor/actuator failure scheduled controller

there exist a structured, bounded, constant feedback matrix C as shown in Figure 1 such that the L_2 into L_2 induced gain from u to y is less than 1? is Π_2 hard.

Proof: We will proceed to convert the generalized quadratic programming problem in (16) to a particular instance of the control problem mentioned. This is done following the same procedure as in [4]. Define the vector $\bar{x} = [.5, .5, \cdots, .5]$ of size $n + m$ and the matrices

$$M = \begin{bmatrix} 0 & 0 & \bar{x} \\ A & 0 & A\bar{x} \\ \bar{x}^t + p^t\bar{x}^t\bar{x}^t A\bar{x} + p^t\bar{x} + c \end{bmatrix},$$

$$\Delta = diag(y_1, y_2, \cdots, y_m),$$

and

$$C = diag(x_1, x_2, \cdots, x_n).$$

Consider the interconnection given in figure 2. Then the gain from u to y will be:

$$|[yx]A[yx]^t + b[yx]^t + c|$$

Then for all y there exists x such that $|[yx]A[yx]^t + b[yx]^t + c| < k$ if and only if, for all Δ there exists C with $|c_i| \leq 1$ such that the gain from u to y in the system of Figure 2 is less than k. The theorem follows. ■

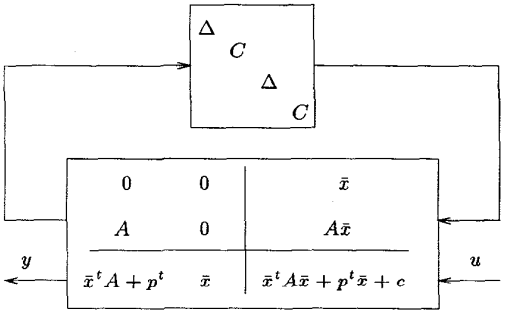

Figure 2: Quadratic programming as a control problem

If we relax the restrictions on the uncertain components and allow them to be operators in L_2, we relax the restriction on the controller by removing the structure and bound conditions, and if we add structure to the system and make the control design problem a full information one, then the preceeding problem becomes the LPV problem described in [9]. This problem can be converted to a convex optimization problem. We can see then how adding structure to the unknown part of the system makes the problem harder, and adding structure to the known part of the system makes the problem easier. These results helps to establish which are the extreme cases on complexity that can happen for similar design questions with different restrictions on the components of the system. The restrictions we had to impose to prove the Π_2 hardness results are extreme. This is in part due to the fact that we are using a binary computation model that brings into the problem artificial restrictions. It is also possible that the Π_2 class is too big for our purposes. There is still no consensus in the computational complexity and numerical analysis communities on which, if any, are the adequate computation models and complexity classes relevant to numerical analysis ([1]). As this debate becomes clearer it will be possible to classify more natural control problems, and get more points in the mapping from structure in uncertainty, controller and system to the computational complexity of the problem.

5 Conclusions

It is becoming increasingly clear that most interesting control problems are naturally hard to solve from a practical point of view. This means that these problems will not accept mathematically elegant solutions. The tools of our trade are going to be computational in nature, and the evaluation of their usability empirical. Although theoretical results are still very much fundamental in the development of the computational tools, it is important to establish precise means to evaluate their contribution to the science.

In this paper we introduce a classification of different computational complexity classes under which control problems fall. We believe that this classification of problems provides one possible gauge to evaluate the importance of theoretical results. Three kind of theoretical results that would be interesting from this point of view are: a result proves a certain problem to be hard in its class, telling us on the real difficulty of its solution; a result proves that a problem is equivalent to another problem in the same class, but better algorithms for the new problem are known; a result can also shown that the result is actually easier than the rest of the class to which it belongs. However, a result stating that a problem in a class is equivalent to another problem in the same class, but that gives no indication that the new problem presents any computational advantage over the first would be deemed incomplete.

The classification of problems according to their complexity is by no means unique. Computational complexity measured with relativized Turing Machines (i.e. the Polynomial Time Hierarchy) tells only part of the story. This classification, and the evaluation of results that results from it, should be taken as one more gauge to help us in the task of sorting out the vast amounts of work carried out in this field.

We also showed in this paper that some two quantifier problems can be proven hard in the corresponding class of the hierarchy. Although the problem presented is highly structured, we believe that some of the restrictions can be lifted and still preserve the hardness of the problem. By comparing this result to previously known result on related problems, we can extend to two quantifier problems

the general classification that has been established in one quantifier problems: more structure in the known parts of the system makes control problems easier; more structure in the uncertain (i.e. undermodelled) or unknown (i.e. to be designed) part of the problem make it harder from a computational point of view.

Acknowledgements

The authors are very grateful to Felipe Cucker for his assistance with the technical details of the Polynomial Time Hierarchy theory, and for explaining us the subtleties of the Blum Shub Smale computation model. This work was partially supported by NASA, NSF and AFOSR.

References

[1] *Panel Discussion: Does Numerical Analysis need a model of Computation*, AMS Summer Seminar: Mathematics of Numerical Analysis: Real Number Algorithms, August 1995.

[2] J.L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*, volume 11 of *Monographs on Theoretical Computer Science*. Springer-Verlag, 1988.

[3] V. Blondel and J. Tsitsiklis. NP-Hardness of some linear control design problems. In *Proceedings of the 34^{th} Conference on Decision and Control*. IEEE, 1995.

[4] R. D. Braatz, P. M. Young, J. C. Doyle, and Manfred Morari. Computational complexity of μ calculation. *IEEE Transactions on Automatic Control*, 39(5):1000–1002, 1994.

[5] M.R. Garey and D.S. Johnson. *Computers and intractability : a guide to the theory of NP-completeness*. A Series of books in the mathematical sciences. W.H. Freeman, 1983.

[6] J. Hartmanis, editor. *Computational Complexity Theory*, volume 38 of *Proceedings of Symposia in Applied Mathematics*. American Mathematical Society, 1989.

[7] A. Megretski and S. Treil. Power distribution inequalities in optimization and robustness of uncertain systems. *Journal of Mathematical Systems, Estimation and Control*, 3(3):301–319, 1993.

[8] K.G. Murty and S.N. Kabadi. Some np-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39:117–129, 1987.

[9] A. Packard. Gain scheduling via linear fractional transformations. *Systems & Control Letters*, 22(2):79–92, 1994.

[10] F. Paganini and J.C. Doyle. Analysis of implicitly defined systems. In *Proceedings of the 33^{rd} Conference on Decision and Control*, pages 3673–3678. IEEE, 1994.

[11] A. Poolla and A. Tikku. On the time-complexity of worst-case system-identification. *IEEE Transactions On Automatic Control*, 39(5):944–950, 1994.